Jean-Daniel wrote in his paper: “Visual Analytics relies on three essential layers: *visualization* to provide effective communication and exploration capabilities for humans to **reason** about the information extracted from the data, *analytics* to **simplify and summarize** the data to be exploitable by humans, to extract information from raw data, as well as to provide some guidance on tis exploration, and *data management* to **store, retrieve, and distribute** the data efficiently to the other two layers.” Data management serves for visual analytics (VA) focusing on supporting exploration and interaction in efficient ways. When facing new complex data, VA allows humans to conduct overviews, summaries and drilling down. And he also mentioned a technical difficulty: managing the latency of queries. When an analyst has a hypothesis in mind and queries the system, an answer should arrive in a few seconds; otherwise the analyst may forget the hypothesis and need to retrieve it when the results arrive. Currently, however, most database queries produce just the opposite; results that are entirely accurate but not bounded in time.

VA involves using faster, less accurate mechanisms to provide initial results quickly and improve the accuracy later if needed. Recently, data management and analysis tools have begun to converge using multiple technologies including parallel computing, cloud computing, and GPGPU. Unfortunately, exploration has not been taken into account in these new infrastructures.

Martin L. Kersten of MonetDB proposed a similar idea about data management in his paper “The Researcher’s Guide to the Data Deluge: Querying a Scientific Database in Just a Few Seconds”(VLDB 2011). The next generation database systems should *interpret queries by human’s intent*, rather than for complete and correct answers. A scientist can stepwise explore deeper and deeper into the database, and stop when the result content and quality reaches his satisfaction point. For this vision, he listed five concrete tracks of potential ground-breaking research:

- One-minute database kernels for real-time performance.
- Multi-scale query processing for gradual exploration.
- Result-set post processing for conveying meaningful data.
- Query morphing to adjust for proximity results.
- Query alternatives to cope with lack of providence.

By the first track, the kernel should identify and avoid performance degradation points on-the-fly and to answer part of query within strict time bounds. Alexander Kalinin implements an interactive data exploration system, and he facilitated a cost model to evaluate each query. This paper was accepted by SIGMOD 2014. I think it is a demo of Martin L. Kersten’s idea, and will be introduced in details later.

The second track is to aim for a *staging scheme*, where stepwise a larger portion of the database becomes the query target. We can break a query for large data into multiple small pieces $Q = Q_1 \cup Q_2 \cup \ldots \cup Q_n$, such that one piece can be evaluated within the bounds of the user budget. The key challenge is to find the cost-metrics and database statistics that allow the system to break queries into multiple steps. Alexander Kalinin use GRID By clause for splitting the search space.

The third track is a visualization work. Some statistical techniques can be processed in linear time over the result set to show the data distribution. For a large result set, sampling technology can be used to reduce processing time.

By the fourth track, query morphing is a kind of prefetching technology. It works as follows, the user gives a starting query Q and most of the effort T is spent on finding the “best” answer for Q. But a small portion is set aside for the following exploratory step. The query is syntactically adjusted to create variations Qi, i.e. with a small edit distance from Q, as shown in Figure 1. Alexander Kalinin designed a heuristic online search algorithm to prefetch data in neighborhood.

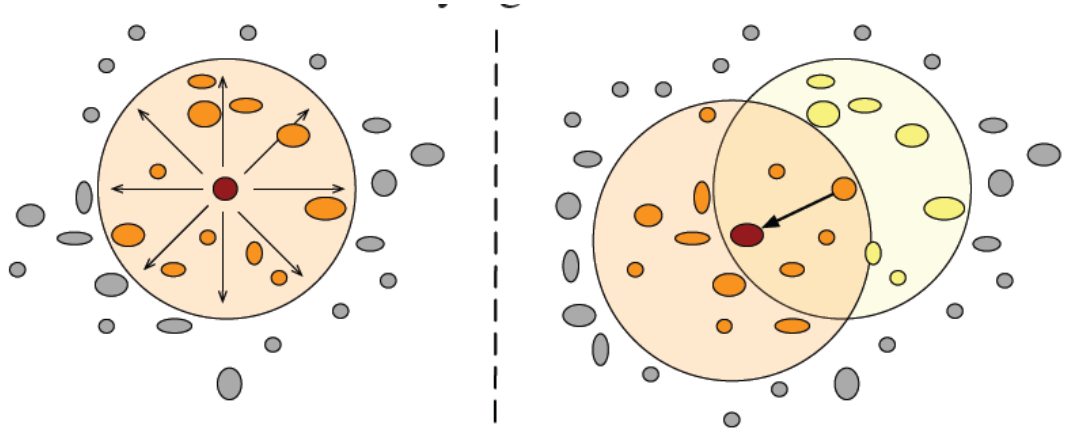

Figure 1: An example of query morphing

The last track means queries as answers, that is to say, return users query suggestions for more effective exploration. There are two crucial research challenges here. First, we should be able to identify “bad” or “wrongly formulated” queries. Second, we need to identify “interesting” queries to return as an answer. For the first part, a bad query can be identified by the optimizer provided good statistical information is available. For the second part, query logs may be a good way to aid in creating the query advisory list.

After reading these surveys, I realized that we can take two steps to complete the data management system for our analysis tasks on mobile data. First, we will optimize the search engine kernel to keep each query in an acceptable time. This work should be executed on a cloud computing platform, and it is not our main effort. Second, we will aim to interactive query on mobile data. We will extract some specific query modes, which is not supported by existing databases, to extend SQL language and meet our tasks’ requirements. Last week, some works of the first step have been carried out. For example, I installed an in-memory database MonetDB and import some mobile data to compare performance with the Hive query system done by Ma. A comparative result will be shown quickly. For the current data size, in-memory database will be the better.

In the next week, I will continue the first step work to optimize on Spark. There is another job to analyze the taxi data based on complex network theory. Prof. He gave us some suggestions in the perspective of economics if some traffic accidents happen. Gu and I realized this direction may be a good option.